CLAIMS

We claim:

1. A static data flow analysis method comprising:

chasing a data flow instance through a data flow graph until a transition instruction is encountered;

resolving the transition instruction to a procedure pointed to by a call graph; and

chasing the data flow instance into a data flow graph of the procedure.

2. The method of claim 1, further comprising:

encountering a pointer dereference operand while chasing through a data flow graph;

chasing backward to resolve where the pointer points; and

continuing chasing the data flow instance from the resolved pointer dereference operand.

3. The method of claim 1 wherein the procedure transition instruction is a call instruction, and the data flow instance chase is a forward chase.

4. The method of claim 1 wherein the procedure transition instruction is a first instruction of a procedure, and the data flow instance chase is a backward chase.

5. The method of claim 1 wherein the data flow graphs contain pointers to an internal representation of a program.

6. The method of claim 5 wherein the internal representation comprises a graph data structure.

7. The method of claim 1 wherein the call graph contains pointers to an internal representation of a program.

8. The method of claim 2 wherein the resolved pointer dereference is a global type, and chasing continues at plural instructions that reference the global as operands.

9. The method of claim 2 wherein the resolved pointer dereference is a field reference type, and chasing continues at plural instructions that reference the field reference as operands.

10. The method of claim 1, wherein a state machine directs data flow chase through alternating states comprising instruction change states and data transformation states.

11. The method of claim 1 wherein the data flow graph of the procedure is built after the transition is resolved to the procedure.

12. The method of claim 1 wherein the inputs to the method comprise binary code, and a start state comprises a data instance and an instruction address in the binary code.

13. A method comprising:

receiving binary code and a start state;

creating from binary code a procedures and instructions representation;

creating a call graph comprising pointers to procedures in the procedures and instructions representation;

creating a data flow graph for a procedure containing the start state, the data flow graph comprising pointers to instructions in the procedures and instructions representation;

chasing a data instance of the start state through instructions in the data flow graph corresponding to states in a state machine; and

upon encountering a procedure transition instruction in the data flow graph, corresponding to a state in the state machine representing the call graph, transitioning the data instance chase to a data flow graph of a procedure identifiable in the call graph.

14. The method of claim 13 further comprising:

upon encountering a pointer dereference in an instruction in a data flow graph corresponding to a state in the state machine representing a pointer dereference table, performing a backward recursive search indicated in the pointer dereference table according to the addressing mode of the pointer dereference, and identifying a location in the backward recursive search.

15. The method of claim 14 wherein the location indicates a field reference definition, and the data instance chase resumes at an instruction indicated by a field reference list.

16. The method of claim 14 wherein the location indicates a global reference definition, and the data instance chase resumes at an instruction indicated by a global reference list.

17. A computer readable medium comprising instructions for performing the method of claim 13.

18. A computer readable medium having instructions for performing a method comprising:

chasing a data flow instance through a data flow graph until a transition instruction is encountered;

resolving the transition instruction to a procedure pointed to by a call graph; and

chasing the data flow instance into a data flow graph of the procedure.

19. The computer readable medium of claim 18 further comprising:

encountering a pointer dereference operand while chasing through a data flow graph;

chasing backward to resolve where the pointer points; and

continuing chasing the data flow instance from the resolved pointer dereference operand.

20. The computer readable medium of claim 18 further comprising:

the procedure transition instruction is a call instruction, and the data flow instance chase is a forward chase.

21. A computer-based service comprising:

means for creating an internal representation of a program;

means for creating a data flow graph comprising pointers to instructions in the internal representation;

means for creating a call graph comprising pointers to procedures in the internal representation; and

means for creating a field reference list comprising pointers to field references in the internal representation.

22. The computer-based service of claim 21 wherein the internal representation is a list data structure.

23. The computer-based service of claim 21 wherein the internal representation is a tree data structure.

24. The computer-based service of claim 21 wherein the internal representation is a graph data structure.

25. The computer-based service of claim 21 wherein the data flow graph edges are bidirectional.

26. The computer-based service of claim 21 wherein the call graph edges are bidirectional.

27. A computer system including a processor and memory, the memory comprising:

a component for receiving binary files and creating internal representations;

a component for accessing internal representations and creating a call data structure;

a component for accessing internal representations and creating a data flow data structure, and

a component for accessing internal representations and creating a global reference data structure.

28. The computer system of claim 27 wherein the memory further comprises a component for accessing internal representations and creating a field reference data structure.

29. The computer system of claim 27 wherein the call data structure and the data flow data structures are graph data structures.

30. The computer system of claim 28 wherein the global reference data structure and the field reference data structure are list data structures.